Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2007, Article ID 60834, 7 pages
doi:10.1155/2007/60834

Research Article

Priority-Based Heading One Detector in H.264/AVC Decoding

Ke Xu, Chiu-Sing Choy, Cheong-Fat Chan, and Kong-Pang Pun

Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong

Received 11 July 2006; Accepted 31 January 2007

Recommended by Jarmo Henrik Takala

A novel priority-based heading one detector for Exp-Golomb/CAVLC decoding of H.264/AVC is presented. It exploits the statistical distribution of input encoded codewords and adopts a nonuniform partition decoding scheme for the detector. Compared with a conventional design without power optimization, the power consumption can be reduced by more than 3 times while the performance is maintained and the design hardware cost does not increase. The proposed detector has successfully been verified and implemented in a complete H.264/AVC decoding system.

1. INTRODUCTION

The Moving Picture Experts Group and the Video Coding Experts Group (MPEG and VCEG) have jointly developed a new video coding standard named as H.264/AVC [1]. Compared with previous coding standards like MPEG-2 or H.263, it achieves nearly the same video quality (by means of PSNR and subjective testing) while requiring 60% or less of the bit rate [2]. This substantial improvement comes at a price of extraordinarily huge computational complexity and formidable memory access, which in turn incur greater power consumption.

On the other hand, CMOS technology has now entered the "power-limited scaling regime," where power consumption moves from being one of many design metrics to be number one design metric. The H.264/AVC processing demands much greater power than MPEG-2 or H.263 due to increased complexity. Therefore, its power consumption should be carefully managed to meet power budget, especially for applications on portable devices. Although power dissipation can be substantially reduced through technology scaling, where designers switch to a smaller geometry to implement the same circuit, power reduction through proper design techniques is more flexible and extensive, especially where geometry scaling is not applicable.

H.264/AVC standard defines a hybrid block-based video codec, which is in general similar to early coding standards, but the important changes occur in the details of each functional block with many new coding techniques. One of these techniques occurs in entropy coding, where two methods, Exp-Golomb for syntax elements above the slice layer and CAVLC (context-adaptive variable-length coding) for quantized transform coefficients, are supported in the baseline profile [3]. During the decoding process, all the Exp-Golomb coded syntax elements require the identification of the position of the first appeared "1" inside each codeword. For CAVLC decoding, some parameters like TotalCoeff, level_prefix, and total_zeros tables [1] also need to identify this first "1" before lookup table operation happens.

Conventional detectors usually are not aware of power consumption. One such example is described in [4] which splits the 16-bit input into 4 parts (4-bit vectors), each of which detects whether there is a "1" among the four input bits. Then these results will determine which part should be further tested. Although the method works well, it is not a power-efficient technique since it treats all the 16 input bits with equal importance. The power consumption bears no relationship with the occurrence of any codewords; no matter how likely they will occur.

General low-power design techniques have been developed for many years. Besides these general methods, video decoding presents a unique power optimization opportunity due to temporal, spatial, and statistical redundancies in digital video data. In this paper, we mainly utilize statistical redundancy during video decoding. A data-driven priority-based heading one detector is proposed, which detects the heading "1" in a bitstream that is organized in 16-bit units. The key idea of our proposal is to exploit the statistical characteristics of the heading one position among the various codewords. A nonuniform decoding scheme is designed

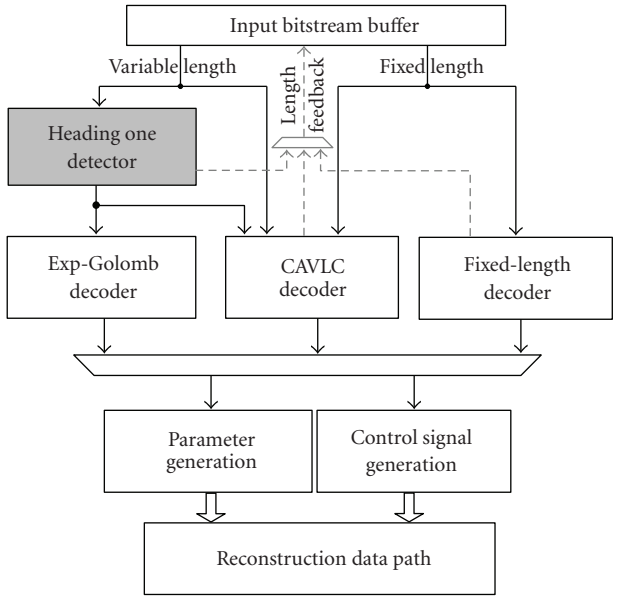

Figure 1: Decoder system architecture.

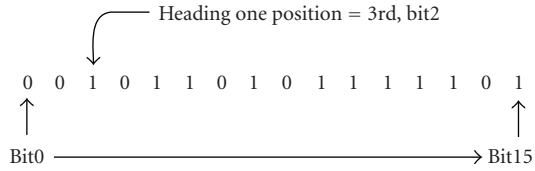

Figure 2: Heading one position.

Table 1: Exp-Golomb codewords.

Code_num	Codeword
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001
...	[M zeros][1][INFO]

accordingly. By selectively disabling some subblocks, the detector consumes much less power without noticeable performance degradation and even smaller design area.

2. BACKGROUND

In this section, we firstly give a brief introduction of the whole decoder architecture. Then we discuss the structure of Exp-Golomb code and CAVLC code which requires heading one detection. At last, we evaluate the related research works in literature.

2.1. H.264/AVC decoding

A simplified system architecture of the whole decoder is illustrated in Figure 1. According to input codeword type, the heading one detector is invoked when current codeword is Exp-Golomb coded or a certain part of CAVLC codewords. Based on the output of the heading one detector, Exp-Golomb codes are mapped from bitstream form to signed, unsigned, or truncated syntax element values, while CAVLC codes are indexed for several lookup Tables (LUT). There is a length feedback signal from the heading one detector, CAVLC decoder, and fixed-length decoder to the input bitstream buffer. The signal indicates how many bits are consumed for decoding current codeword. According to decoded codewords, related parameters and control signals are generated to orchestrate the following reconstruction data path.

2.2. Heading one detection

Figure 2 depicts a normal input to the heading one detector, where the detector needs to search among the 16 bits to find the first appeared "1." Here we assume the input bitstream is encoded from left to right. This example indicates that the heading one position lies at third place (bit2). Although there are several "1's" at some other positions like bit4, bit5, and so forth, they are not heading ones.

Exp-Golomb codes

Exponential Golomb codes (see [5]) are variable-length codes with simple and regular structure as depicted in Table 1. One does not need to store the conversion table for the purpose of decoding, since the correspondence between symbols and codes is mathematically defined. The leading M zeros, as well as the middle "1," are treated as "prefix" of the codeword, while INFO, which is equal in length to the M zeros, is called "suffix" [6]. In Table 1, the first code_num "0" does not contain any leading zero or trailing INFO. Code_nums "1" and "2" have a single-bit leading zero and corresponding single-bit INFO field, code_nums 3~6 have a two-bit leading zeros and INFO field, and so on. Theoretically the codeword table can be infinitely extended according to the coding rule described. The length of each Exp-Golomb codeword is $(2M+1)$ bits long and each codeword can be inferred by the following equation [6]:

$$\begin{aligned} M &= \text{floor}(\log_2[\text{code\_num}+1]), \\ \text{INFO} &= \text{code\_num}+1-2^M, \end{aligned} \tag{1}$$

where floor (x) is a function finding the largest integer which is less than or equal to x.

In H.264/AVC standard, there are three types of Exp-Golomb coding: unsigned, signed, and truncated. They all follow the same coding rule and are only different in whether an additional "code_num to syntax_value" mapping is needed.

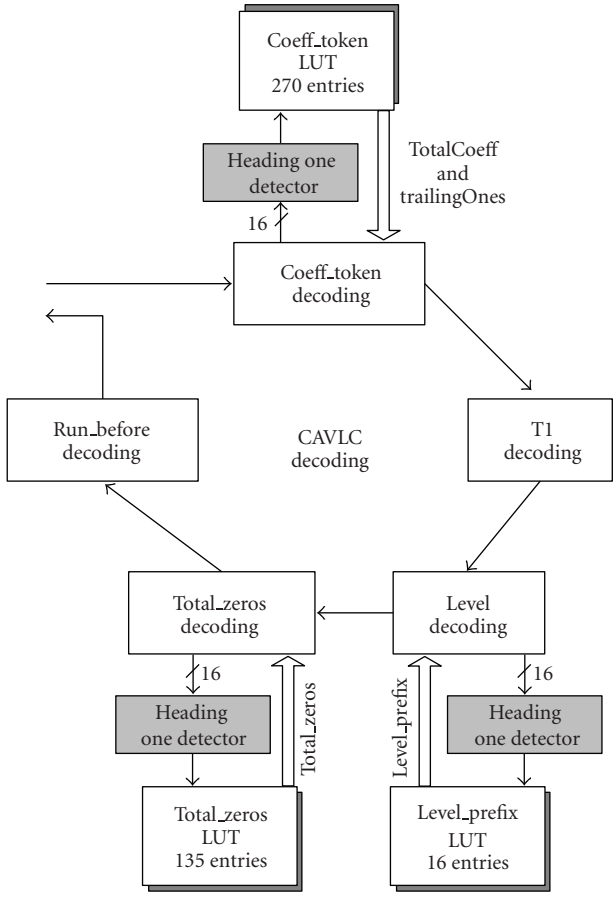

Figure 3: CAVLC decoding flow.

Table 2: Codeword table for level_prefix.

level_prefix	Bit string
0	**1**
1	0**1**
2	00**1**
3	000**1**
4	0000 **1**
5	0000 0**1**
6	0000 00**1**
7	0000 000**1**
...	...

CAVLC

A more efficient algorithm for transmitting the quantized transform coefficients is proposed in [3]. In this method, VLC tables for various syntax elements are selected depending on already transmitted syntax elements. To decode the indexes for some of these VLC tables, a heading one detector is indispensable. The CAVLC decoding step is briefly described in Figure 3.

The CAVLC decoding can be partitioned into five steps and three of them require heading one detection.

Table 3: Total_zeros table for 4×4 blocks with TotalCoeff (coeff_token) 1 to 3.

Total_zeros	TotalCoeff (coeff_token)		
	1	2	3
0	**1**	**1**11	0**1**01
1	0**1**1	**1**10	**1**11
2	0**1**0	**1**01	**1**10
3	00**1**1	**1**00	**1**01
4	00**1**0	0**1**1	0**1**00
5	000**1** 1	0**1**01	00**1**1
6	000**1** 0	0**1**00	**1**00
7	0000 **1**1	00**1**1	0**1**1
...	...	...	...

Table 2 shows one VLC table [1] in CAVLC codes which maps input bit stream to "level_prefix." The value of level_prefix is directly determined by the position of the first appeared "1." Table 3 shows another VLC example where finding the heading one position is sufficient for the whole syntax element to be extracted.

Since most of the syntax elements are coded either as Exp-Golomb codes or CAVLC codes, heading one detector is used extensively in H.264/AVC decoding.

2.3. Related works

Although there are some designs in literature dealing with Exp-Golomb or CAVLC decoding [4, 7–9], few of them mentioned how heading one detection was realized. The only reference design is found in [4]. It proposed a detector that evenly splits the input into four subwords. From each subword, the presence of "1" is detected. Then these results will determine which subword should be further tested, as shown in Figure 4. Priority encoder0's output indicates the position of "1" in the subword, while priority encoder1's output indicates which subword has the heading one. In fact, this is a two-level encoder and cannot run in parallel. Encoder1 selects a subword based on priority where part [3 : 0] has the highest priority and part [15 : 12] has the lowest priority. According to encoder1's indication, encoder0 chooses one correct subword among the four and encodes the heading "1" in the chosen subword as the final heading one position. No matter where the heading one is, four subword decoders and two priority encoders are active all the time.

3. PROPOSED ARCHITECTURE

In this section, we firstly explore the heading one statistics in entropy coding. Based on the observation, a priority-based heading one detector is then proposed.

3.1. Characteristic of entropy coding

As aforementioned, design in [4] proposed a "first 1 detector" based on a uniform input bit-vector partition. That is an effective scheme but no power optimization was considered.

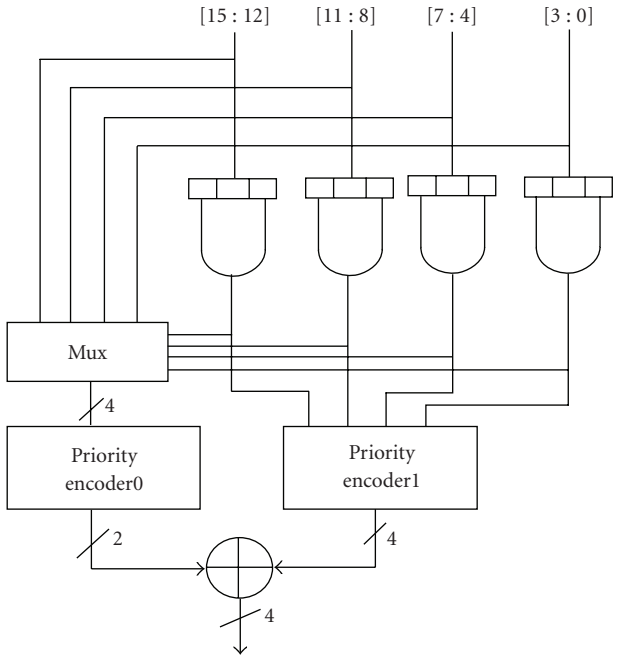

Figure 4: Evenly partitioned detector in [4].

Table 4: Statistic result of heading one position (nearly 0 means that percentage is less than 0.01%).

Position	Whole input bitstream	Intracoded frame	Intercoded frame
0	55.36%	51.37%	56.61%
1	24.15%	24.36%	24.08%
2	11.16%	10.70%	11.31%
3	5.49%	6.21%	5.26%
4	2.17%	3.69%	1.72%
5	0.88%	1.6%	0.65%
6	0.41%	0.91%	0.25%
7	0.16%	0.4%	0.08%
8	0.08%	0.25%	0.03%
9	0.06%	0.24%	0.01%
10	0.04%	0.15%	Nearly 0
11	0.01%	0.06%	Nearly 0
12	Nearly 0	0.04%	Nearly 0
13	Nearly 0	0.03%	Nearly 0
14	Nearly 0	Nearly 0	Nearly 0
15	Nearly 0	Nearly 0	Nearly 0
Average	0.81	1.12	0.74

Since both Exp-Golomb and CAVLC codings are entropy coding methods, they have the same important characteristic like all other entropy coding schemes: shorter codewords are assigned to symbols that occur with higher probability, whereas longer codewords are assigned to symbols with less frequent occurrences. In an H.264/AVC bitstream, the longest code is 16 bits including the heading "1." However, the average length of such kind of codes is not $(16 + 1)/2 = 8.5$, but much smaller.

3.2. SystemC modeling

In order to study the entire bitstream parsing process where entropy decoding is included, we developed a high-level systemC model, emulating the control and communication of real video decoding. Its output is compared with JM9.4 software [10] to verify correct function. The systemC model has internal counters to count the total number of Exp-Golomb codes and CAVLC codes which require heading one detection. It also has individual counters for the number of these codes under different heading one positions. Five popular test videos, named as container, foreman, akiyo, news and carphone, with QCIF 300 frame sequences at 30 fps are used. They are encoded by JM software with quantization parameters set to 22, 25, 28, 32, and 36, respectively. The statistical profile of heading one's positions was hence obtained from simulation with these input bitstreams.

The average codeword lengths are found as in Table 4 (note that if a "1" is in the first bit, this corresponds to position = 0 and so on). The intraframe and interframe have slightly different heading one statistical position percentage since usually the intraframe has more residual information and needs more CAVLC decoding effort. For example, inside interframe, positions equal to or above 10 begin to have nearly zero (less than 0.01%) codes distribution, whereas for intra frame, this boundary is pushed to a high position which indicates that only positions 14 and 15 have nearly zero codes distribution. However, both intra- and interframes-share the same tendency that the higher the position is, the less opportunity that a heading one is found.

Be aware that the statistical positions stated in Table 2 are not a simple average of the values in the intra- and the interframe columns. This is because intra- and interframes have different total numbers of Exp-Golomb/CAVLC codes in different test video sequences. For example, in akiyo video sequence of 300 frames, 24% of codes need heading one detection are extracted from intraframes and 76% are extracted from interframes, while in foreman video sequence, 30% of these codes are extracted from intraframes and 70% are from inter frames. In addition, distributions of heading one positions (position $= 0, 1, 2, \ldots$) in a single video sequence vary from one bitstream to another. These nonuniform codewords distributions lead to the nonlinear relationship of total average positions for intra- and interframes. In addition, positions of interframes tend to have a larger weight than those of intraframes in all the video sequences tested, for there are more intercoded frames than intra-coded ones.

According to Table 4, the heading one in a codeword is located on average in a position indicated in Figure 5. We conclude that the average heading one position for the whole sequence/intraframe/interframe is 0.81/1.12/0.74, respectively, which are much smaller than the simple average of 8.5. Of course, positions naturally are whole numbers, fractional values are the artifacts of averaging.

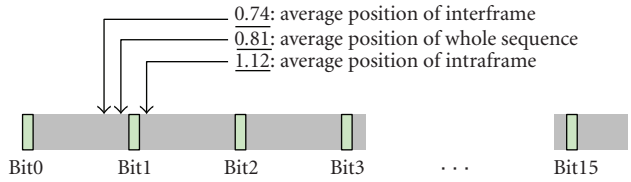

Figure 5: Average heading one position.

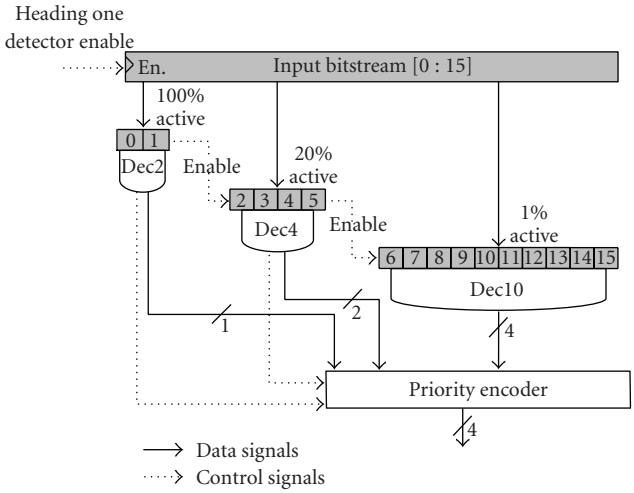

Figure 6: Proposed heading one detector.

3.3. Proposed architecture

From the above analysis, we conclude that the position of heading one most likely lies around the second input bit (position = 1). The first two positions (position0 + position1) account for almost 80% of all cases and the first six positions (position0 + · · · + position5) account for nearly 99%. Thus we propose a priority-based nonuniform partition heading one detector where the input 16 bits are divided into 3 unequal subdetectors and each subdetector can be selectively enabled and disabled. Figure 6 shows the proposed scheme.

In our design, input bitstream from bitstream buffer is controlled by an "enable" signal. If current codeword needs heading one detection, the whole 16 bits are enabled and passed to the heading one detector, else the detector is disabled to reduce unnecessary switching. The entire detector is partitioned into three parts, each of which handles a different chunk of input bits with varying priority. Dec2, which has the highest priority, processes the first two input bits and is active all the time to detect whether there is a "1" and its corresponding likely position (only first bit position or second bit position here). If a "1" is found in dec2, which signals a successful identification of the heading "1" in a codeword, the position information is passed to the final priority encoder to generate a heading one position. At the same time, the lower-priority dec4 and the lowest-priority dec10 are disabled to save power. Conversely, there is a 20% possibility that dec2 will fail to find a "1" and dec4 will be enabled thereafter. More rarely, both dec2 and dec4 cannot find a heading one and dec10 will then be active but having only 1% possibility. The outputs of dec2, dec4, and dec10 are selectively encoded as heading one position of whole 16-bit input by a priority encoder.

Design in [4] divides 16-bits input evenly into 4 identical subwords. Each subword decoder detects whether there is a "1" inside and their outputs are then sent to two priority encoders. No matter whether the "1" found in each subword is a heading "1," all four subword decoders, as well as the priority encoders, are active all the time. However, if the first decoder which looks at bits [3 : 0] finds a "1," no matter what the outcome of the other three decoders is, one can conclude that the first "1" is in bits [3 : 0]. The work done by the other three decoders is of no consequence and only a waste of power.

4. DESIGN ANALYSIS

In this section, we mainly discuss and compare the power consumption of [4] and the proposed design. We also discuss the speed and area overheads.

4.1. Theoretical analysis

Strictly speaking, power consumption constitutes of dynamic power and static power. Since the target process is a relatively standard CMOS 130 nm technology and the circuit is small enough, the static power only contributes a very small portion of the whole power consumption. Therefore, we can assume that the detector's entire power is proportional to dynamic power to facilitate our calculation. Average power dissipation for decoding each heading one position can be modeled as suggested in [11]

$$E_{\text{avg}} = \sum_{i=1}^{N} P_i E_i, \tag{2}$$

where P_i is the probability that heading position $= i$ will occur, E_i is the energy required to detect such a position, and N is the total number of possible positions where $N = 16$ for H.264/AVC codes.

Since dynamic power consumption is almost linear to the complexity of these decoding units, without loss of generality, one can assume the power consumed by dec2 is 2 units, dec4 is 4 units, and dec10 is 10 units. In [4], all the four decoders are identical and consume 4 units of power all the time.

Estimated power consumption for the detector in [4] is:

$$E_{\text{avg}} = \sum_{i=1}^{4} P_i E_i = 4 \times 100\% \times 4 \text{ units} = 16 \text{ units}. \tag{3}$$

In our scheme, three decoders are active sequentially and their activation rate is proportional to the heading one distribution shown in Table 4.

Table 5: Layout power analysis.

	Power consumption at 20 MHz real-time QCIF/30 fps		
Frame type	Implementation of [3]	Our proposal	Power reduction
Intra frame	13.45 μW	3.99 μW	3.38 times
Inter frame	2.35 μW	0.733 μW	3.21 times

Table 6: Physical implementation.

Technology	UMC 130 nm
Metal layer	6 metals, 2 thick
Supply voltage	1.08 v
Max. frequency	200 MHz

Estimated power consumption for our schemeis

$$\begin{aligned} E_{\text{avg}} &= \sum_{i=1}^{3} P_i E_i \\ &= 100\% \times 2 \text{ units} + 20\% \times 4 \text{ units} + 1\% \times 10 \text{ units} \\ &= 2.9 \text{ units}. \end{aligned} \tag{4}$$

The percentages in the above equation reflect the activity rate of each submodule dec2 (for position 0~1), dec4 (for position 2~5), and dec10 (for other positions), respectively. The overhead-like power consumed by muxes is negligible. The relative power saving for our scheme is about 5.5 times while the throughput is nearly the same.

4.2. Implementation analysis

Since there is no power consumption figures reported in [4], to have a fair comparison, we built a "heading 1 detector" according to [4] with the same process technology used for our scheme. Both of the detectors are integrated into H.264/AVC decoding system, where there is a switch to control which one is currently active. The decoding system is simulated by ModelSim. The Verilog RTL codes are then synthesized by design compiler and are placed and routed by Astro. Parasitic information is extracted by Star-RCXT and postsimulation is processed in VCS. Based on the layout database and individual activity rate obtained from post-sim, postlayout power analysis results can be obtained from PrimePower, shown in Table 5. The key implementation parameters of our scheme are listed in Table 6. Considering that the heading one detector has the highest switching activity in entropy decoding, the power reduction contributable by such a detector is substantial.

According to Tables 5 and 6, one can conclude that our design not only consumes less power, but is capable of performing real-time decoding. The circuit size is even a little bit smaller than the design in [4]. Although a larger dec10 is introduced, two priority encoders found in [4] are reduced to one which leads to slight area reduction. The only penalty is a small throughput degradation if the heading one happened to be at a higher position like 6, 7, and so forth, because dec2, dec4, and dec10 will need to be triggered in sequence to obtain the final result. Even at this extreme case, the proposed design can achieve a maximum frequency of 200 MHz, which is substantially faster than other building blocks in the whole H.264/AVC decoding system.

The advantage of our design is drawn from exploiting the high probability of "heading one" lying in the first few bits of a codeword. By using a nonuniform decoding structure, a lot of power is saved because one does not need to search all bits. The same technique can also be applied to other entropy decodings such as that in MPEG-2. Although the codeword structure is not identical as in H.264/AVC, short codewords inherently occur more frequently. Proposed technique can then be employed according to the specific statistical profile found from high-level modeling.

5. CONCLUSION

A priority-based, data-driven power-efficient heading one detector has been proposed. The opportunity to reduce power is identified at architectural level through systemC modeling. Appropriate circuit implementation is then chosen. It exploits the statistical codeword distribution of an entropy-coded bitstream, and a novel power-saving decoding scheme is subsequently devised. Compared with conventional detectors, the proposed design achieves more than 3 times power reduction while maintaining area and speed performance. It does not utilize any special techniques such as clock gating or voltage scaling, and thus makes it readily employable in other circumstances when different technologies may be used. Since power consumption in ICs is a critical issue in recent years, this paper suggests an effective method to reduce power by exploiting statistical characteristics.

ACKNOWLEDGMENT

The work reported is supported by a Hong Kong SAR Government Research Direct Grant no. 2050322.

REFERENCES

[1] J. V. Team, "Advanced video coding for generic audiovisual services," *ITU-T Recommendation H.264 and ISO/IEC 14496-10 AVC*, May 2003.

[2] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 688–703, 2003.

[3] T. Wiegand, G. J. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[4] W. Di, G. Wen, H. Mingzeng, and J. Zhenzhou, "An Exp-Golomb encoder and decoder architecture for JVT/AVS," in *Proceedings of the 5th International Conference on ASIC*, vol. 2, pp. 910–913, Beijing, China, October 2003.

[5] S. W. Golomb, "Run-length encoding," *IEEE Transactions on Information Theory*, vol. 12, no. 3, pp. 399–401, 1966.
[6] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression*, John Willey & Sons, New York, NY, USA, 2003.
[7] Joint Video Team (JVT) reference software JM9.4, http://iphome.hhi.de/suehring/tml/download/.
[8] T.-C. Wang, H.-C. Fang, W.-M. Chao, H.-H. Chen, and L.-G. Chen, "An UVLC encoder architecture for H.26L," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 2, pp. 308–311, Phoenix, Ariz, USA, May 2002.
[9] S. H. Cho, T. Xanthopoulos, and A. P. Chandrakasan, "A low power variable length decoder for MPEG-2 based on nonuniform fine-grain table partitioning," *IEEE Transactions on VLSI Systems*, vol. 7, no. 2, pp. 249–257, 1999.
[10] I. Amer, W. Badawy, and G. Jullien, "Towards MPEG-4 part 10 system on chip: a VLSI prototype for context-based adaptive variable length coding (CAVLC)," in *Proceedings of IEEE Workshop on Signal Processing Systems (SIPS '04)*, pp. 275–279, Austin, Tex, USA, October 2004.
[11] H.-Y. Lin, Y.-H. Lu, B.-D. Liu, and J.-F. Yang, "Low power design of H.264 CAVLC decoder," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 2689–2692, Island of Kos, Greece, May 2006.